形式语言与自动机项目报告

轨道交通连锁系统建模

|  |  |
| --- | --- |
| 姓 名： | 陈麒先 |
| 学 号： | 2001213040 |
| 学 院： | 信息科学技术学院 |
| 任课教师： | 金芝 教授 |

2021年1月

郑重声明

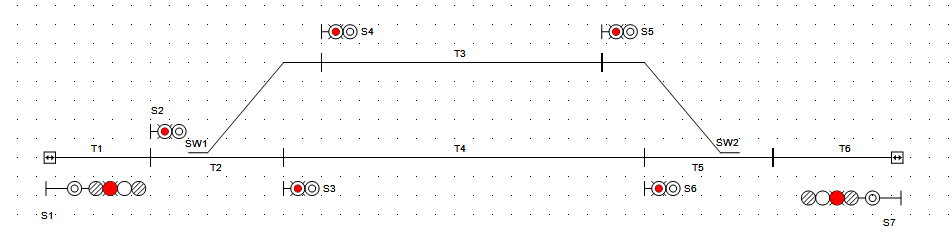
**本项目由作者独立完成。抄袭行为在任何情况下都是不能容忍的(COPY is strictly prohibited under any circumstances)！由抄袭所产生的一切后果由抄袭者承担，勿谓言之不预也。**

**陈麒先**

一、系统设置的自然语言描述

1、系统场景的自然语言描述

一个铁路可以被划分为铁路编组站和连接两个编组站的轨道线路。一个铁路编组站由四种组件构成：轨道区段（Railway），道岔（Switch，又叫Point），进路（Route），信号灯（Signal Light 或者Light）。连锁系统由于要通过的进路不同，使得系统内的对象个数会不一样。比如对应如图所示的站场，它的进路就有两个。一个是上面一条路，一个是下面一条路。具体走哪个，是在头一开始就设置好的。



上图为站场展示图。该车站有两个道岔(分别为SW1和SW2)、七个信号灯(分别为S1至S7)、六条轨道段(分别为T1至T6)。至于确定好走哪条进路，开哪些灯，开哪些道岔，都是由进路表决定。进路表如下表所示。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Route | | | Signal lights | | Points | | Railway |
| ID | From | To | Green | Red | Open | |  |
| Up | Down |
| R1 | S1-S7 | | S1,S2,S4,S5,S7 | S3,S6 | SW1,SW2 |  | T1,T2,T3,T5,T6 |
| R2 | S1-S7 | | S1,S2,S3,S6,S7 | S4,S5 |  | SW1,SW2 | T1,T2,T4,T5,T6 |

2、运行场景的自然语言描述

本系统实际的轨道交通联锁系统的用例场景，对于火车进入轨道的处理流程的实例如下：

* 当一列火车进入轨道之前，火车Train向控制器Center发出请求。列车Train等候红绿灯信号，若收到红灯red信号则列车等待并重复发送请求；若收到绿灯信号green，则列车Train进入轨道，发出trainEnter信号。
* 控制器Center收到列车Train发出的请求信号request后，Center向轨道Track发送检查是否占用信号checkoccupied。
* 轨道Track收到检查是否占用信号checkoccupied后，根据全局变量y(轨道占用与否的变量)判断轨道上是否有列车，若y=1表示轨道被占用，则发出占用信号occupied；若y=0表示轨道未被占用，则发出未被占用信号unoccupied。当列车进入轨道后，判断变量置为1，当列车离开轨道后，判断变量置为0。
* 控制器Center收到占用信号occupied后回到初始状态；收到未被占用信号unoccupied后发出dolock信号将要使用的道岔锁住。
* 道岔Point初始状态为解锁状态unlocked，收到控制器发出的锁住信号dolock后进入锁住状态locked；列车离开后控制器发出解锁状态后，道岔由locked锁住状态变为unlocked解锁状态。
* 控制器发出道岔封锁信号dolock后发出信号灯变绿信号dogreen。
* 信号灯Light初始状态为红灯RED，收到控制器发出的变绿信号dogreen后变为绿灯状态GREEN，并发出绿灯信号green；列车离开后控制器发出变红信号dored后，信号灯由绿灯状态变为红灯状态。
* 控制器控制道岔、信号灯做出相应改变后，接受到列车进入信号trainEnter和列车驶出信号trainLeave；接收到该信号后发出道岔解锁信号dounlock和信号灯变红信号dored，接收到red信号后结束整个流程。

3、系统实体描述

根据上述系统场景和运行场景，可以为目标系统建模设计出以下六个实体：

* 调度器: 调度器负责调度火车进入轨道连锁系统以及在整个为火车选择合适的轨道行驶。
* 控制器: 接收火车、轨道、信号灯、道岔发来的各种信号，并控制这些实体进行相应的状态变化，完成火车进入轨道、离开轨道的一次调度活动。
* 火车: 可以向控制器发送请求进入某段轨道的信号以及列车进出轨道的信号，状态可以分为未进入连锁系统、已进入连锁系统和处于哪条进路这几种。
* 铁轨: 火车在持续行进过程中的一段铁轨，可以感应到列车是否行驶在轨道上，因此状态有空闲和被占用两种；轨道可以向控制器发送其状态信息，以辅助控制器决定是否允许火车通行。
* 道岔:道岔是一种使列车从一条轨道转入另一轨道的线路连接设备;列车要转换轨道前需要锁住相应的道岔，完成轨道转换后解锁道岔，因此道岔的状态设置为上锁与解锁。
* 信号:信号灯根据轨道的状态发出红绿灯信号，其中红灯代表禁止列车通过，绿灯代表允许火车通行。

本系统将针对上述实体进行建模。

4、系统参数设定

本课题根据信号、铁轨、道岔和火车数量不同，共设有两种难度：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 问题难度 | Signal light个数 | Point个数 | Track个数 | Train的个数 |
| 1 | 1 | 1 | 1 | 1 |
| 2 | >=2 | >=2 | >=2 | >=2 |

本项目则采用样例提供的轨道图进行系统建模，根据样例设置，本项目为系统设定了参数如下：

|  |  |  |  |
| --- | --- | --- | --- |
| Signal light个数 | Point个数 | Track个数 | Train的个数 |
| 7 | 2 | 6 | 3 |

二、预备知识

1、项目背景介绍

轨道交通是指运营车辆需要在特定轨道上行驶的一类交通工具或运输系统。最典型的轨道交通就是由传统火车和标准铁路所组成的铁路系统。随着火车和铁路技术的多元化发展，轨道交通呈现出越来越多的类型，不仅遍布于长距离的陆地运输，也广泛运用于中短距离的城市公共交通中。

轨道交通连锁系统的任务是保证行车安全、协调列车运行、提高运输效率。铁路或地铁车站以及车辆段都有很多线路，线路的两端以道岔连接，根据道岔的不同位置组成列车的不同进路，每条进路只允许一列列车使用。

列车能否进入某进路，是否会发生进路冲突，这些都由联锁系统来协调。联锁是“通过技术方法，使信号、道岔和进路必须按照一定程序并满足一定条件，才能动作或建立起来的相互关系”。也就是说，为了保证车站行车安全，必须制定一系列联锁规则以制约信号的开放与关闭、道岔转动和进路的建立；必须以技术手段来实现这些联锁规则。联锁系统以电气设备或电子设备实现联锁功能，以信号机、动力转辙机和轨道电路室外三大件来体现联锁功能。

为了保证列车的正常运行、人员的生命安全，该系统对实时性、可靠性的要求较高，需要借助形式化的方法对系统进行建模设计，并使用相关的模型检测工具进行严格的性质验证。因此，本项目具有较大的研究价值和现实意义。

2、时间自动机

时间自动机是一套对实时系统进行建模和验证的理论。这一理论是Alur和Dill的杰出工作成果。时间自动机就是带有时钟集的有限自动机。时钟集是有限个时钟的集合，每个时钟都是一个取值范围为0或正数的变量。时间自动机状态之间的转换要满足时钟约束才可能发生。时间自动机的状态可以附加上“位置不变性”的属性，这也是一个时钟约束，用来保证状态不会保持在原地不动。时钟的存在使得时间自动机可以对一些时间约束进行建模，增强了对实时系统的建模表达能力。

本项目将以时间自动机理论为基础，采用uppaal工具做为建模与验证工具进行实现。

3、Uppaal工具介绍

UPPAAL是一个集成的工具环境，被用来对被转换时间自动机网络模型的实时系统进行建模、校验和验证。它是由瑞典Uppsala 大学的信息技术学院和丹麦Aalborg大学的计算科学学院联合开发的。截止到2010年官方的发布版本为 Uppaal 4.0.13。以学术为目的的话，该工具可以免费获取，若是商用的话则需要支付一定的费用。

该工具可在下载UPPAAl安装包，安装UPPAAL要求有java8运行环境，UPPAAL4.0版本需要比较高的JDK版本，如1.7，1.8版本，本项目在Mac OSX系统上安装了java8环境及相关jdk。下载好UPPAAL压缩文件之后，解压后会得到一个uppaal.jar文件，安装该文件就可以安装好UPPAAL。uppaal.jar是一个可执行的jar包，在控制台运行命令：java -jar uppaal.jar即可成功启动并运行UPPAAL。

本项目使用 UPPAAL进行系统的建模以及后续的模型验证。

三、自动机建模

为了得出变迁系统模型，可以利用时间自动机模型对每一个实体进行建模，以下将对每一个系统环境实体（包括火车、铁轨、道岔和信号四项）进行简单的建模描述以及在uppaal中的建模结果。

1、火车

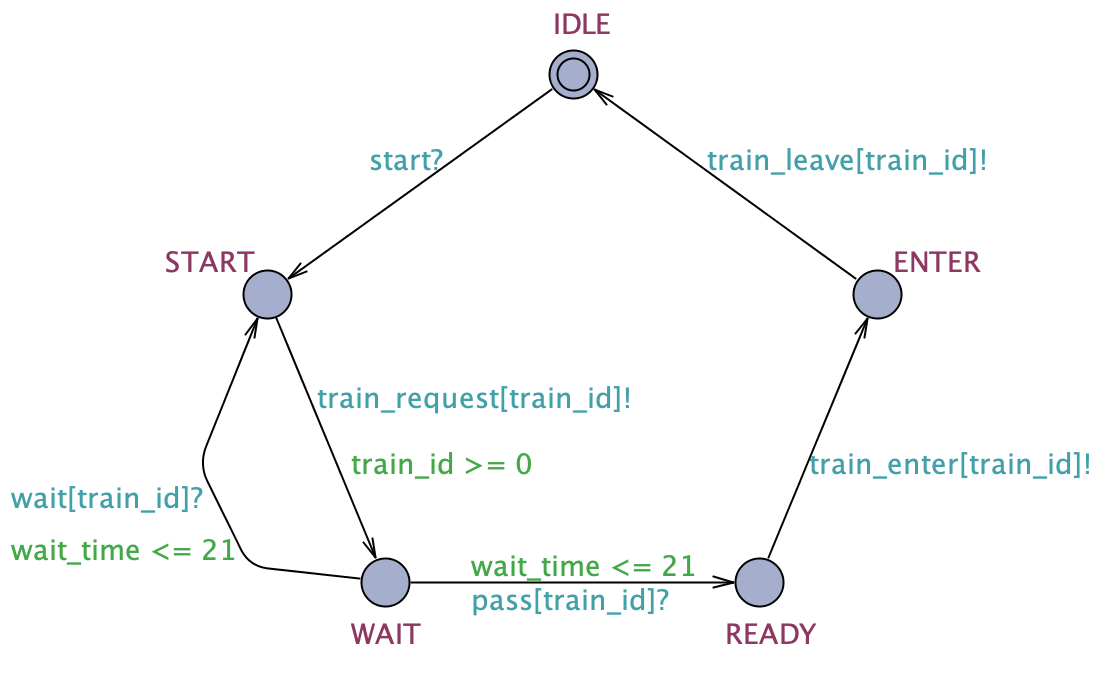
本项目为火车实体设计了4中不同的状态：IDLE、BUSY、UP、DOWN。各状态的含义如下：

* IDLE：火车尚未进入轨道交通系统，即火车目前空闲。
* BUSY：火车已驶入轨道交通系统，即当前处于忙碌状态。
* UP：火车从道岔驶入场景中上面的铁轨。
* DOWN：火车从道岔驶入场景中下面的铁轨。

火车实体命名为train，为其设计的属性如下表所示：

|  |  |
| --- | --- |
| 属性名 | 属性说明 |
| train\_id | 根据本文第一节的设计，系统可能存在多辆火车，该属性表明了每辆火车独一无二的编号。 |
| train\_state | 该属性表示了火车当前所处的状态，与前面的状态定义对应，共有4种可能的取值。 |

火车实体的变迁系统建模结果如下图所示。



火车初始时还未接受调度，火车空闲，处于IDLE状态。当调度器向火车发送start触发信号时，火车状态转换为START启动状态，进入新一轮的调度。随后，火车向控制器发起train\_request[train\_id]的信号，表明train\_id所表示的火车实体对铁轨资源发起请求。接着，火车实体进入WAIT的等待状态，等待控制器后续信号控制。若后续收到控制器发送的通过信号pass，则火车已准备好驶入铁轨，转变为READY就绪状态；若后续收到的等待信号时等待信号wait，则说明当前铁轨已被占用，火车被禁止继续前行，因此应将该火车实体重置为IDLE空闲状态。进入READY状态的火车，可以继续前行，向调度器发送train\_enter信号，进入到ENTER驶入状态，最后，火车会离开当前铁轨，发送train\_leave信号，表示当前火车实体已结束本轮调度，重置为IDLE空闲状态。

按照约束条件，应该满足火车总体等待时间不超过21秒的性质。

2、铁轨

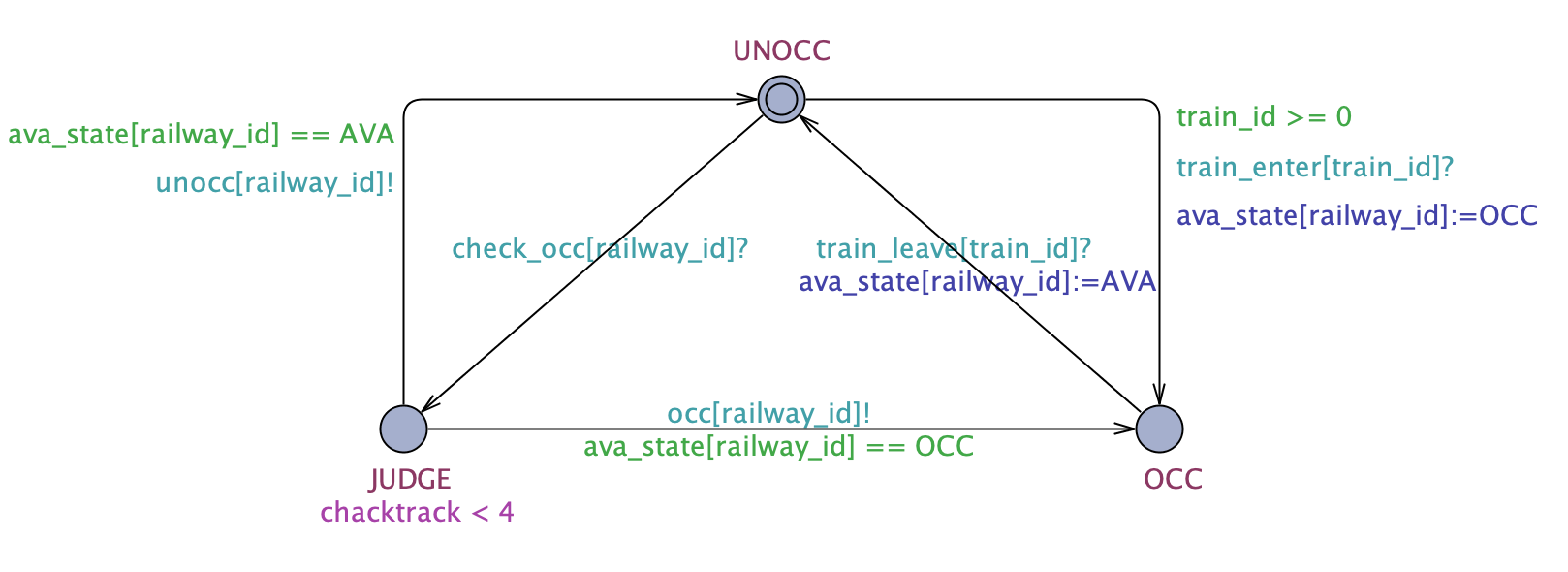
铁轨实体使用railway标识符表示，铁轨共有两种状态：

* AVA：表示当前铁轨是可用的，没有火车占用此铁轨。
* OCC：表示当前铁轨已被占用，是不可用的。

铁轨共有两种属性如下：

|  |  |
| --- | --- |
| 属性名 | 属性说明 |
| railway\_id | 根据本文第一节的设计，系统可能存在多道铁轨，该属性表明了每个铁轨独一无二的编号。 |
| ava\_state | 该属性表示了铁轨当前所处的状态，与前面的状态定义对应，共有2种可能的取值。 |

铁轨实体的变迁系统建模如下：



铁轨实体的初态为UNOCC表示轨道未被占用。当控制器向铁轨发出check\_occ信号时，出发铁轨状态转换，进入到JUDGE判断状态，铁轨实体会根据当前铁轨的占用状态ava\_state判断铁轨是处于AVA可用状态，还是OCC占用状态，进而发出occ或unocc信号。

注意此处JUDGE判断状态持续时间应满足少于4秒的限制。

当铁轨接收到火车发出的train\_enter信号后，将轨道置为占用状态OCC；当铁轨接收到火车发出的train\_leave信号后，将轨道置为可用状态AVA。

3、道岔

道岔实体共有两种状态：

* LOCKED：当前道岔处于锁死状态，说明火车能够安全驶入。
* UNLOCK：当前道岔出入未上锁状态，说明火车无法安全通行。

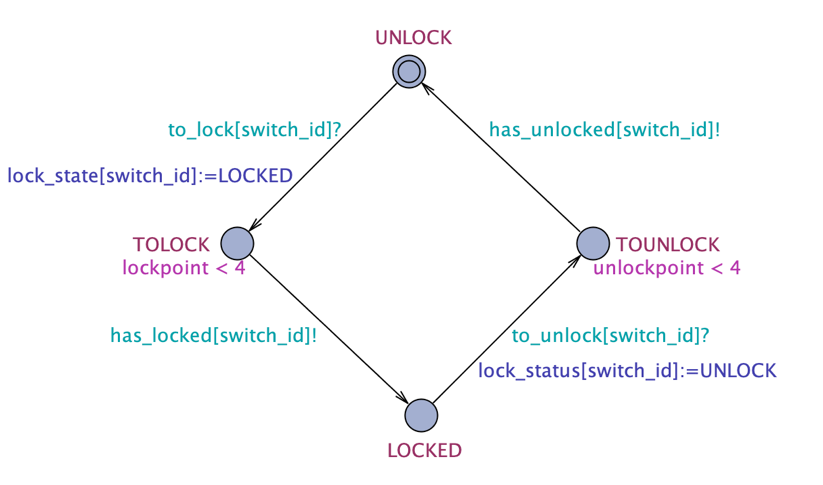
道岔实体用switch标识符表示，本项目为该实体设计了如下的属性：

|  |  |
| --- | --- |
| 属性名 | 属性说明 |
| switch\_id | 根据本文第一节的设计，系统可能存在多个道岔，该属性表明了每个道岔独一无二的编号。 |
| lock\_state | 该属性表示了道岔锁当前所处的状态，与前面的状态定义对应，共有2种可能的取值。 |

道岔实体初始时设置为UNLOCK未上锁状态，当收到控制器的to\_lock信号时触发状态向TOLOCK的转换，表示将为道岔实体进行加锁；加锁完毕时，将由道岔发出has\_locked信号，表明加锁操作已经完成，当前道岔被锁住。当收到来自控制器的to\_unlock信号时触发状态向TOUNLOCK的转换，表示将为道岔实体进行解锁；解锁完毕时，将由道岔发出has\_unlocked信号，表明解锁操作已经完成，当前道岔被解锁。

根据实体性质约束条件，加锁和解锁过程均应严格小于4秒钟。

道岔实体的变迁系统建模结果如下：



4、信号

信号实体共有两种状态：

* GREEN：当前信号处于绿灯亮起状态，说明火车被允许驶入。
* RED：当前信号处于红灯亮起状态，说明火车被禁止驶入。

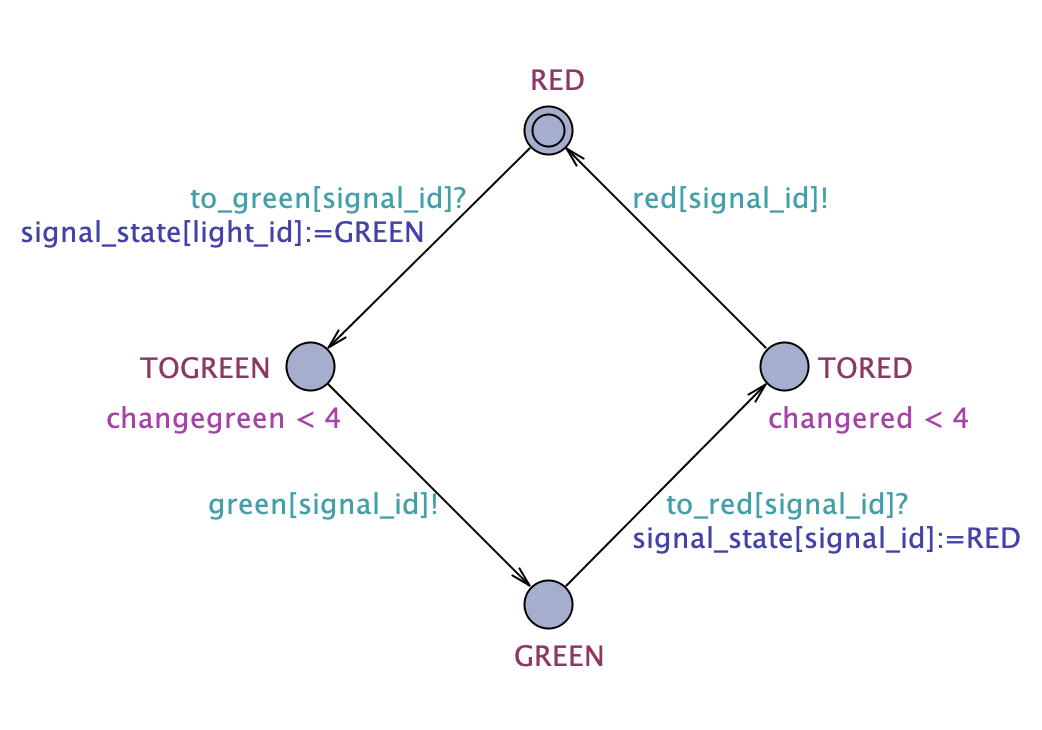
信号实体用signal标识符表示，本项目为该实体设计了如下的属性：

|  |  |
| --- | --- |
| 属性名 | 属性说明 |
| signal\_id | 根据本文第一节的设计，系统可能存在多个信号，该属性表明了每个信号独一无二的编号。 |
| signal\_state | 该属性表示了信号灯当前的亮起状态，与前面的状态定义对应，共有2种可能的取值。 |

信号实体初始时设置为RED红灯亮起状态，当收到控制器的to\_green信号时触发状态向TOGREEN的转换，表示将转换信号至绿灯亮起；信号转换完毕后，将由信号实体发出green信号，表明切换至绿灯亮起动作已经完成，当前绿灯亮起。当收到控制器的to\_red信号时触发状态向TORED的转换，表示将转换信号至红灯亮起；信号转换完毕后，将由信号实体发出red信号，表明切换至红灯亮起动作已经完成，当前空灯亮起。

根据实体性质约束条件，信号灯由绿变红以及由红变绿的过程均应严格小于4秒钟。

信号实体的变迁系统建模结果如下：



四、系统与环境实体交互

本项目中的系统与环境实体之间的交互主要是通过调度器和控制器进行实现的，以下将对系统与环境实体交互的调度器和控制器进行简单的建模描述，以及给出在uppaal中的建模结果。

1、调度器

调度器采用router标识符表示。调度器是整个系统的核心组件，其主要功能为控制火车的一系列调度动作，进而实现多进路轨道交通连锁系统的路由调度功能。

本项目为调度器设计了两个属性：

|  |  |
| --- | --- |
| 属性名 | 属性说明 |
| stage | 调度器的调度是分阶段的，主要的划分依据为整个系统中所含有的铁轨数量。该属性记录了每个调度阶段的调度情况，包括火车是否被调度和哪辆火车被调度。当没有需要被调度的火车时，将该属性置为IDLE表示空闲阶段。 |
| has\_switch | 该属性表示了当前调度的铁轨中是否含有道岔。例如，在单路直铁轨上没有道岔，在分叉道路上存在道岔。因此需要该属性区分本次调度中是否需要考虑道岔的存在。 |

根据建模的连锁系统包含的轨道数目，将整个调度器划分成不同的调度阶段;每个调度阶段负责检查相应的一条轨道将来是否会有火车进入。为了达到多列火车可以同时进入连锁系统这个目的，调度器是按照从进路末端到进路始端这种顺序进行调度的。需要注意的是，调度顺序会在很大程度上影响对问题的建模。可以证明本项目采用的逆向调度不仅可以大大简化模型的设计，而且还具有很好的时序关系，提升整个系统的运行效率。

模型的初始状态为START。从该状态出发，会对整个系统进行一个初始化操作，保证所有的火车都处于还未进入系统的IDLE状态，所有的信号灯处于红灯状态，所有的道岔处于解锁状态，所有的轨道都处于未被占用状态以及调度器的每个调度阶段都处于空闲状态。

stage[6]到 stage[1]状态分别代表调度器正处于对轨道6到1的调度阶段。如果该阶段没有需要调度的火车，则直接跳过该阶段进入下一个阶段的调度状态;否则需要调度某列火车进入相应的轨道，进入该调度阶段的执行状态，即ACTi状态。在完成状态转移的同时，调度器会安排需要调度的火车编号，该火车请求进入的轨道编号、相关的信号灯编号以及确定请求进入的轨道是否包含道岔；如果存在道岔，还需要指定好道岔的编号。

在ACTi状态下，调度器会向发出本轮调度的开始信号start，并随即进入FINi状态，等待本轮调度的完成。当收到控制器发来end信号后，调度器进入ACTi-1状态，转入对下一轮调度的处理。

本项目在建模期间对以下细节进行了处理

1) 火车驶离最后一段轨道后，需要将火车的状态更新为IDLE，表示火车已离开整个连锁系统，并且可以再次被调度进入系统。

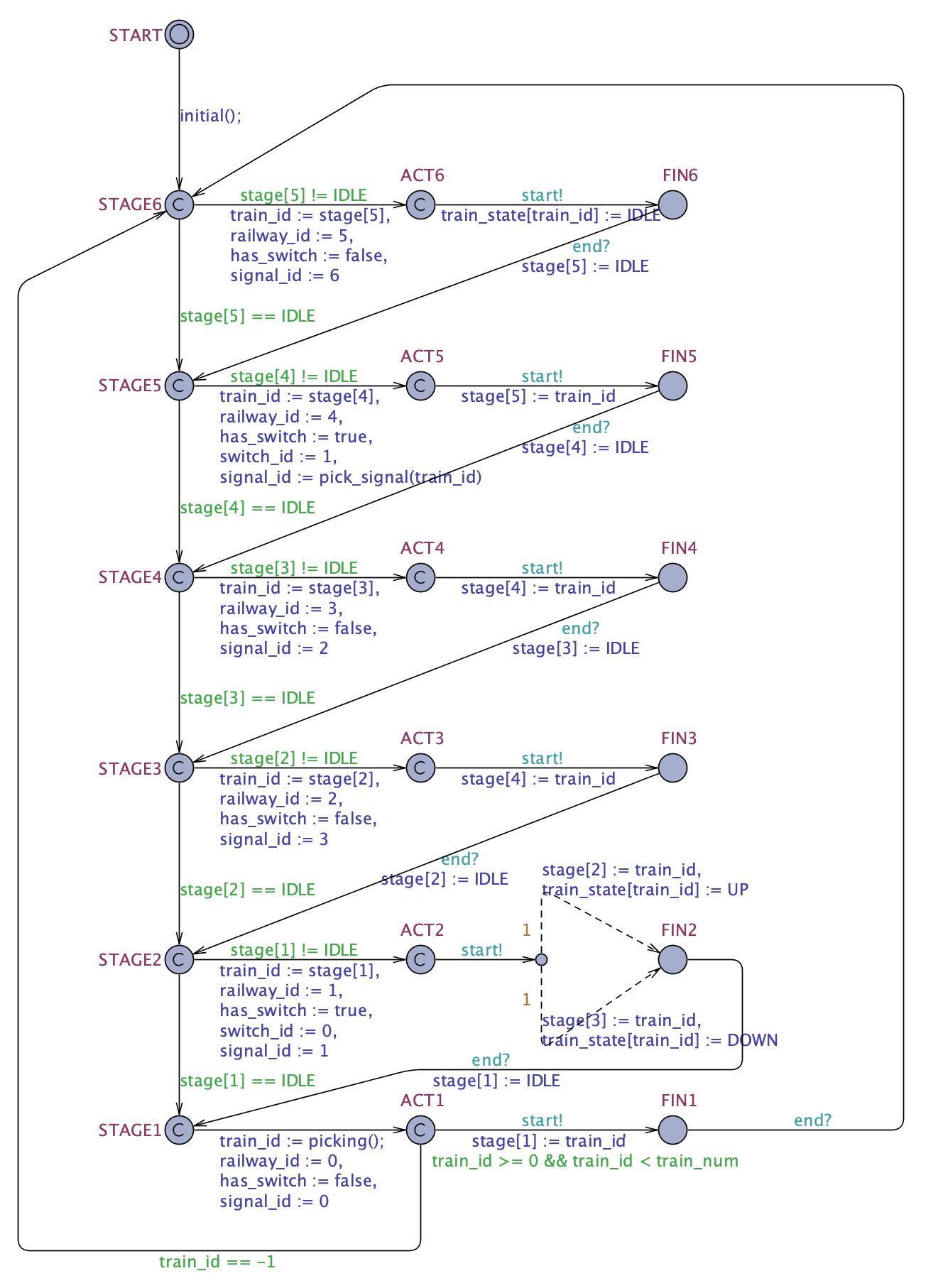
2) 完成一轮调度后，需要将调度器的状态置为IDLE，表示当前的调度阶段已处理完本轮调度，可以重新安排一次新的调度。

3) 铁轨T5是上下两条进路汇合的一条轨道，该轨道上有两个信号灯，分别用于控制上下两条进路上的火车。因此调度器中对应的调度阶段需要根据火车选择的进路确定需要控制哪一个进路上的信号灯。

4) 铁轨T2是上下两条进路分开的一条轨道，调度器需要为请求进入该条轨道的火车选择进路，进而完成路由功能。在我们系统中，调度器以相等的概率1，使每辆进入该条轨道的火车选择上下两条进路。

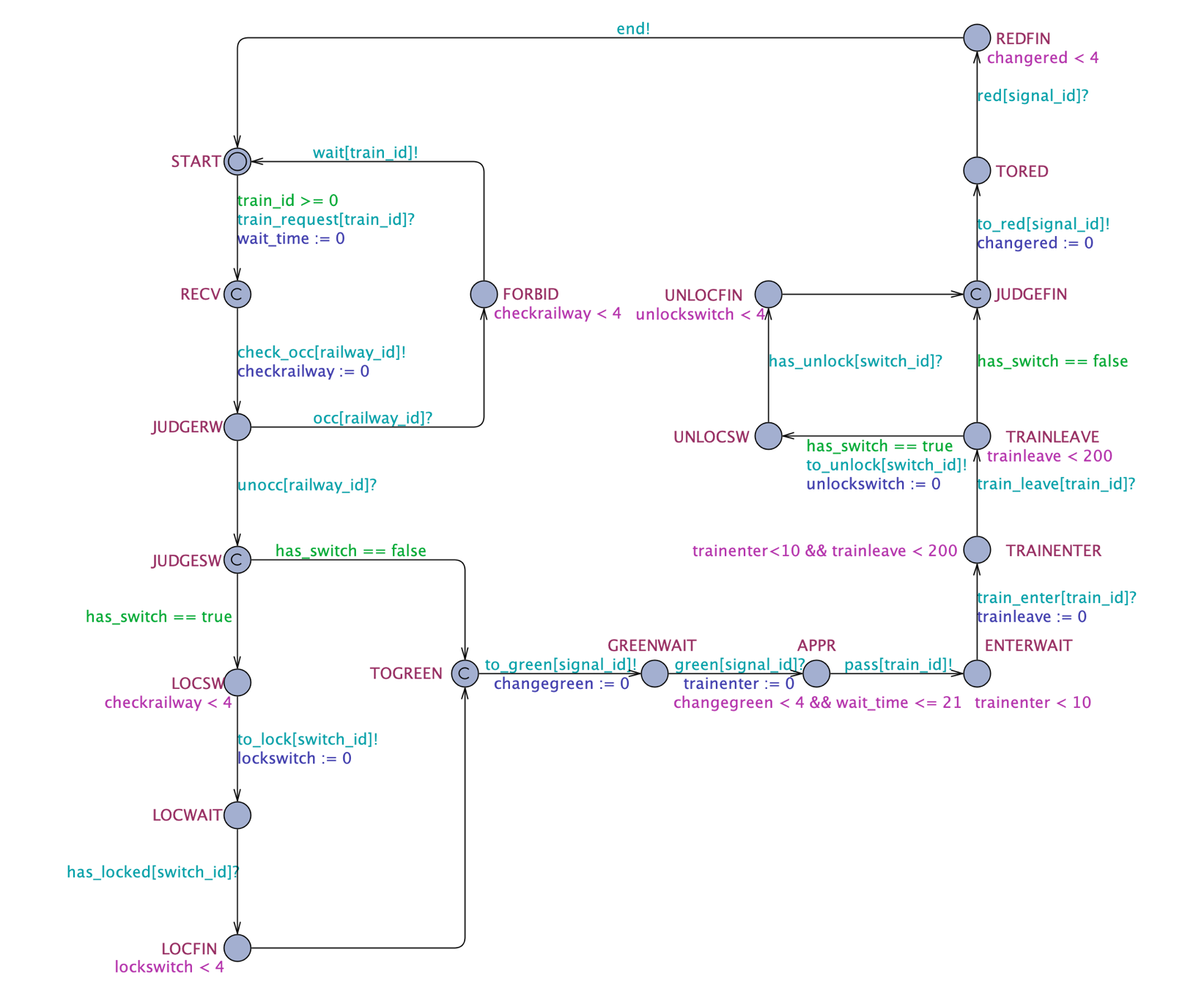
5) 铁轨T1是整个连锁系统的起始轨道，因此调度器需要每次安排一辆处于IDLE状态，即未进入连锁系统的列车并调度其进入轨道T1。如果系统中不存在这样的一辆火车，就跳过该调度阶段。

调度器实体的变迁系统建模结果如下：



2、控制器

控制器用controller标识符表示。下图展示了控制器变迁系统建模结果。



控制器负责协调火车、信号、铁轨和道岔进行一次完整的火车进入轨道然后离开轨道的过程，因此包含的状态较多。

初始状态为 START。当控制器处于该状态时，会一直等待该轮被调度的火车发来的请求进入轨道信号；收到相应的信号后，控制器进入RECV状态。RECV状态表示控制器已经收到火车发来的请求信号，接下来控制器会向轨道系统发送check\_occ信号，进入JUDGESW 状态。JUDGERW状态表示控制器正在检查轨道的占用状态。当收到轨道被占用的信号occ时，通知火车无法前进，发送 wait 信号，结束本次调度并重新回到START状态；如果轨道未被占用就会收到unocc信号，进入JUDGESW状态。

JUDGESW状态表示控制器允许火车进入轨道，但在发出放行信号之前还需完成对相关的道岔和信号灯的控制操作。首先控制器会检查火车请求进入的轨道是否包含道岔，如果有道岔，需要锁定道岔，进入LOCSW状态；如果不包含道岔，则跳过锁定道岔的环节，直接进入对信号灯进行控制的部分，即TOGREEN状态。

LOCSW状态表示控制器当前正在对道岔进行加锁操作。控制器会向对应的道岔 发送 to\_lock信号，并进入LOCWAIT状态。

在LOCWAIT状态，控制器等待道岔系统发来的上锁完成信号has\_locked；当收到加锁完成信号后，随即进入LOCFIN状态，表示道岔加锁操作已全部完成。稍后控制器会进入 TOGREEN状态。

TOGREEN状态表示控制器当前正在控制信号灯进行相应的颜色变化，由红灯变为绿灯。控制器向信号灯系统发送to\_green信号，并进入GREENWAIT状态。

在GREENWAIT状态，控制器等待信号灯系统发来的已变为绿灯的信号green；当收到 green 信号后，控制器进入APPR状态。

APPR状态表示控制器允许列车进入轨道，向列车发送放行信号pass，然后进入 ENTERWAIT状态。

ENTERWAIT状态表示控制器当前正在等待列车发来的进入轨道信号。当接收到 train\_enter 信号后，控制器进入TRAINENTER状态，表示火车已经进入轨道。

控制器在TRAINENTER状态会等待接收来自火车的 train\_leave 信号，收到信号后就进入 TRAINLEAVE状态，表示火车已驶出轨道。

接着，控制器完成后续对道岔和信号灯的恢复操作。类似的，如果轨道不包含道岔，则直接由TRAINLEAVE状态进入JUDGEFIN状态，跳过解锁道岔的操作；否则，会向道岔系统发出 to\_unlock的解锁信号，并进入UNLOCSW状态。

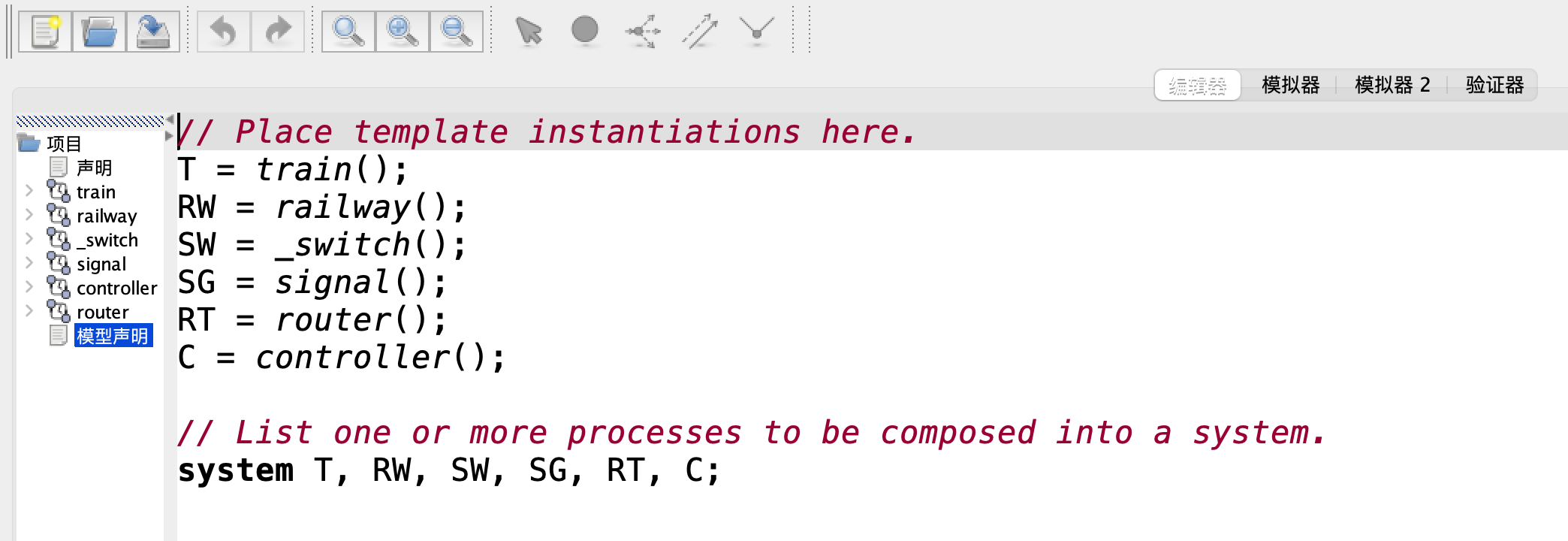
UNLOCSW状态表示控制器当前正在对道岔进行解锁操作，当收到解锁完成信号 has\_unlock，控制器进入UNLOCSW状态，并随即进入JUDGEFIN状态。

在JUDGEFIN状态下，控制器会发出to\_red信号，以控制信号灯重新变为红灯，进入 TORED状态。当收到信号系统发来的 red 信号后，进入REDFIN状态，表示信号已变为红灯。

最后完成上述所有的操作后，控制器会向调度器发送end信号，表示该轮调度已完成，重新回到START状态。

3、模型声明

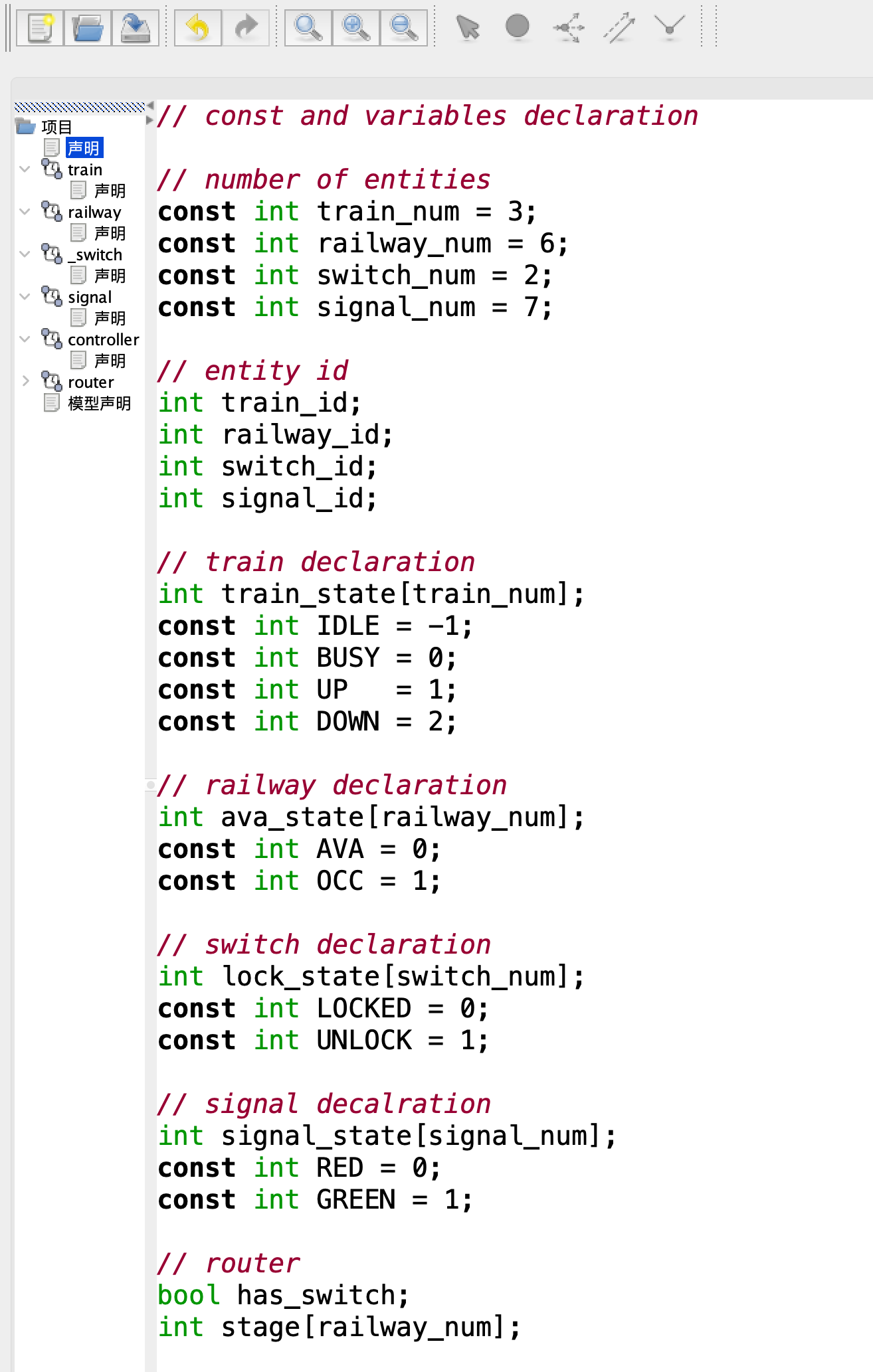
本系统应同时执行6个模型，为6个模型分别实例化，使用system执行如下。



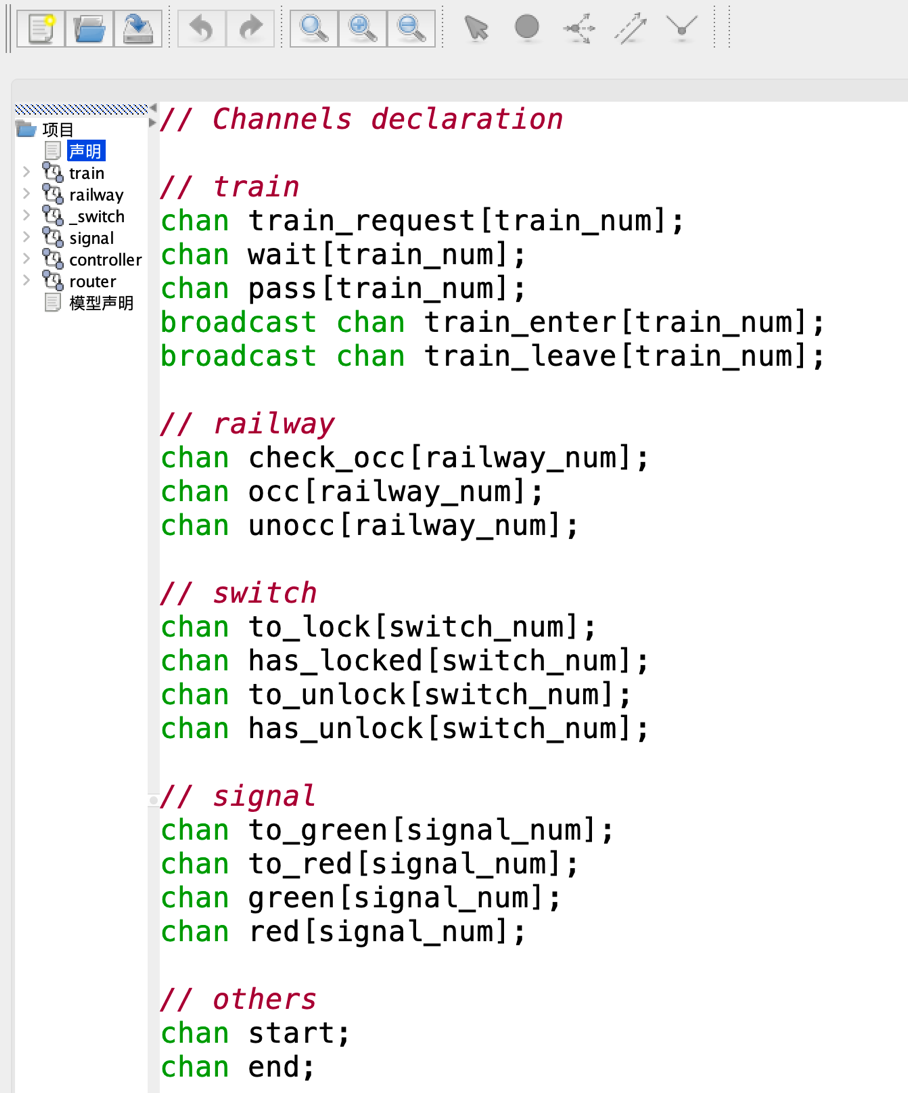
4、全局变量声明

本系统为全局变量声明如下：

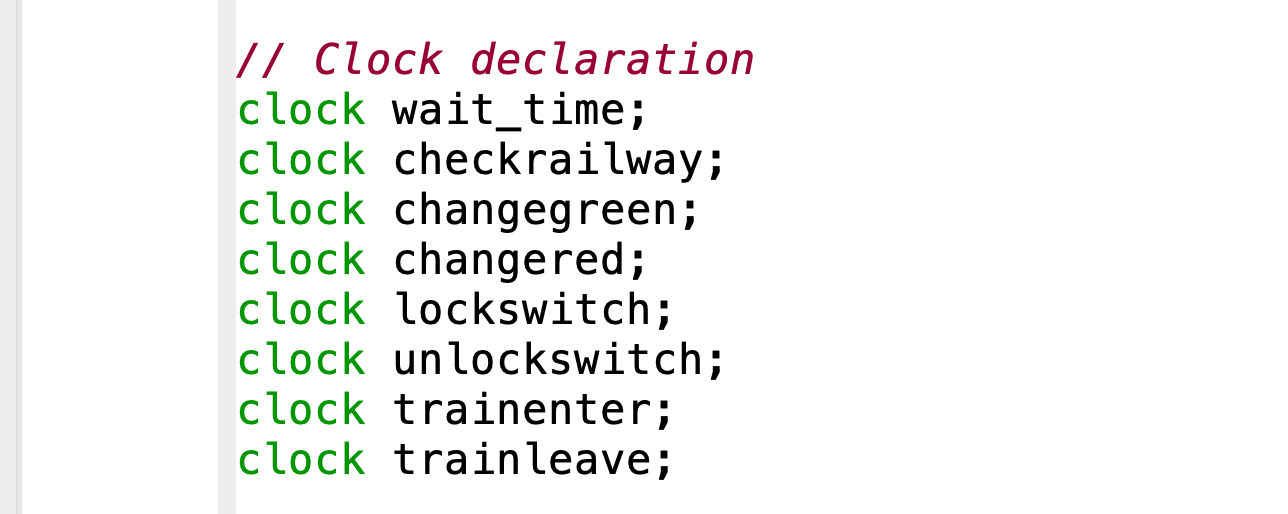
首先声明与实体相关的常数及数组，常数用const表示，状态命名与取值与上文定义相对应。



随后定义各信道名称，用于连接。



最后定义时钟，用于性质验证和不变量控制，如下：



五、系统满足性质定义

根据领域知识，连锁系统必须遵循以下时间约束：

* 控制器模型收到request消息到列车接收到交通灯回馈的消息之间的时间为21s。
* 控制器模型发出checkoccupied消息到收到控制器模型发出反馈消息之间的时间少于4s。
* 控制器模型发出dolock消息到收到locked消息之间的时间少于4s。
* 控制器模型发出dogreen消息到收到green消息时间之间的时间少于4s。
* 控制器模型收到green消息到收到trainEnter消息之间的时间少于10s。
* 控制器模型收到trainEnter消息到收到trainLeave消息之间的时间少于200s。
* 控制器模型发出dounlock消息到收到unlocked消息之间的时间少于4s。
* 控制器模型发出dored消息到收到red消息之间的时间少于4s。

对应的形式化定义的线性时间性质如下：

* A[] C.APPR imply wait\_time <= 21
* A[] C.LOCSW imply checkrailway < 4
* A[] C.LOCFIN imply lockswitch < 4
* A[] C.APPR imply changegreen < 4
* A[] C.TRAINENTER imply trainenter < 10
* A[] C.TRAINLEAVE imply trainleave < 200
* A[] C.UNLOCFIN imply unlockswitch < 4
* A[] C.REDFIN imply changered < 4

另外，还应添加如下性质：

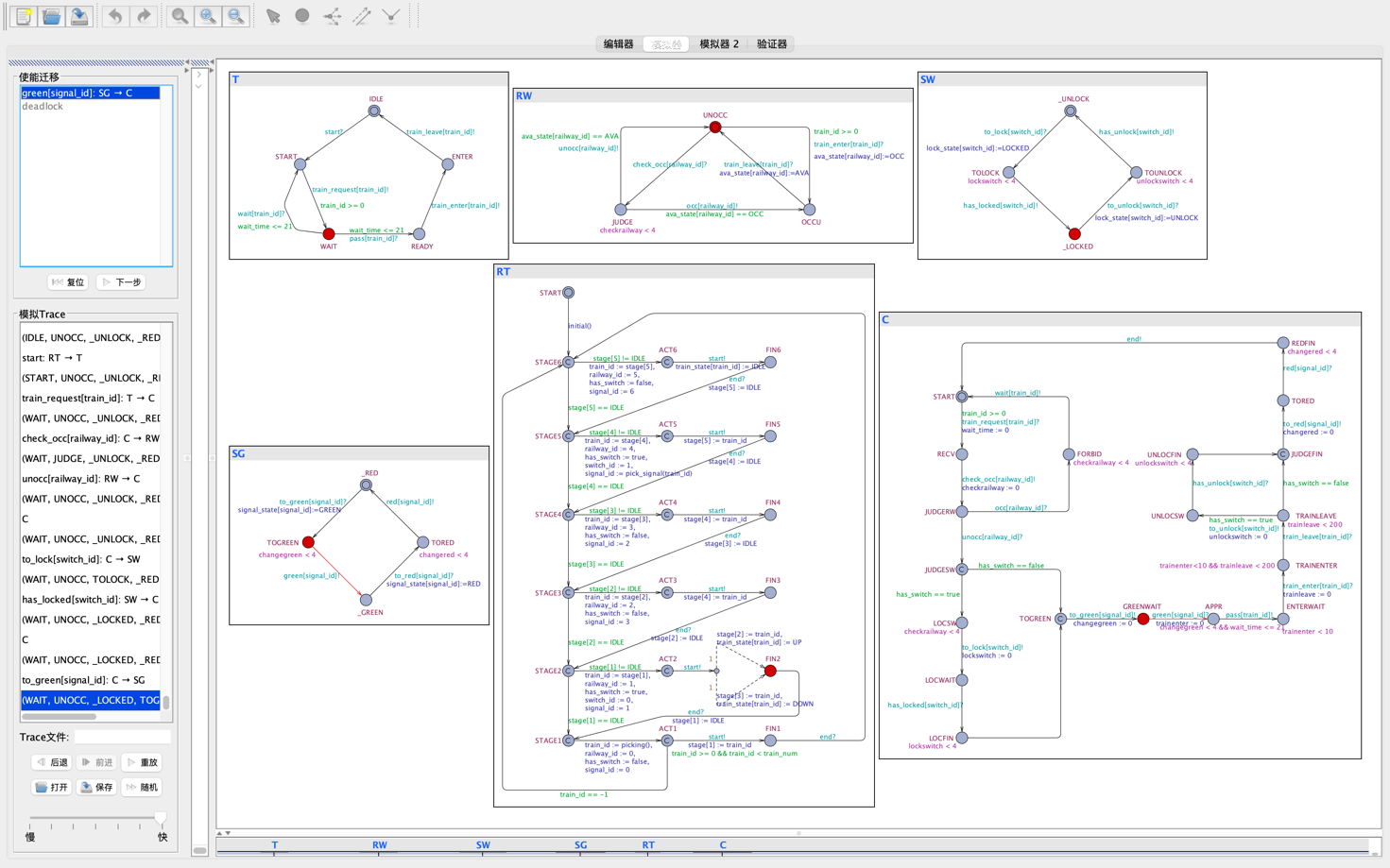
* A<> SG.GREEN imply (RW.UNOCC && SW.UNLOCK)
* A[] not deadlock

上述性质表明，当信号为绿灯时，必用铁轨处于未被使用，且道岔处于未上锁状态。

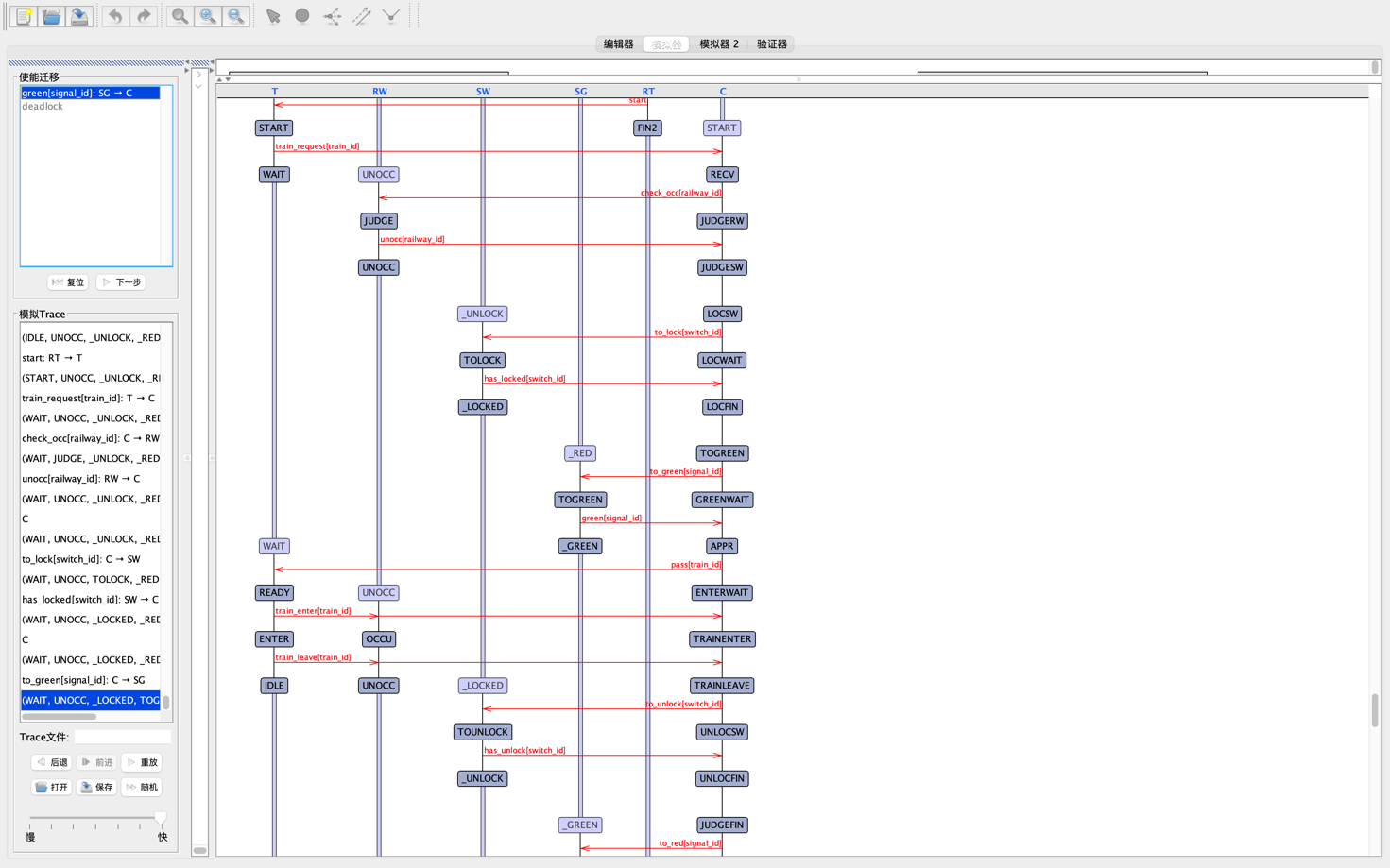
整个系统不应有死锁存在。

六、模型检测结果

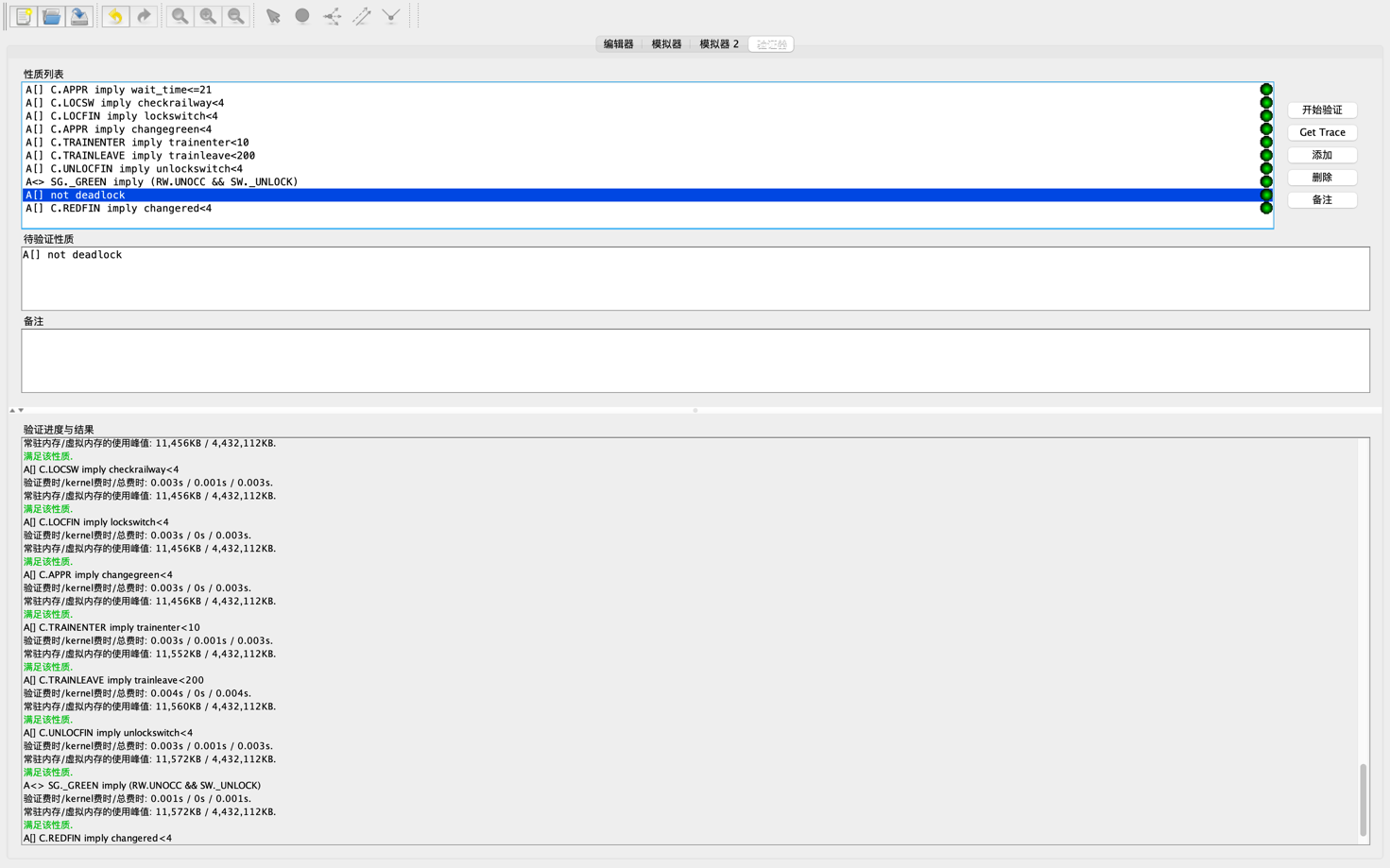
模型整体运行情况某一步的效果，和模型运行时序图截图如下：



从图中可以看出各实体间通过chan进行信号传递，系统各部分存在交互行为。



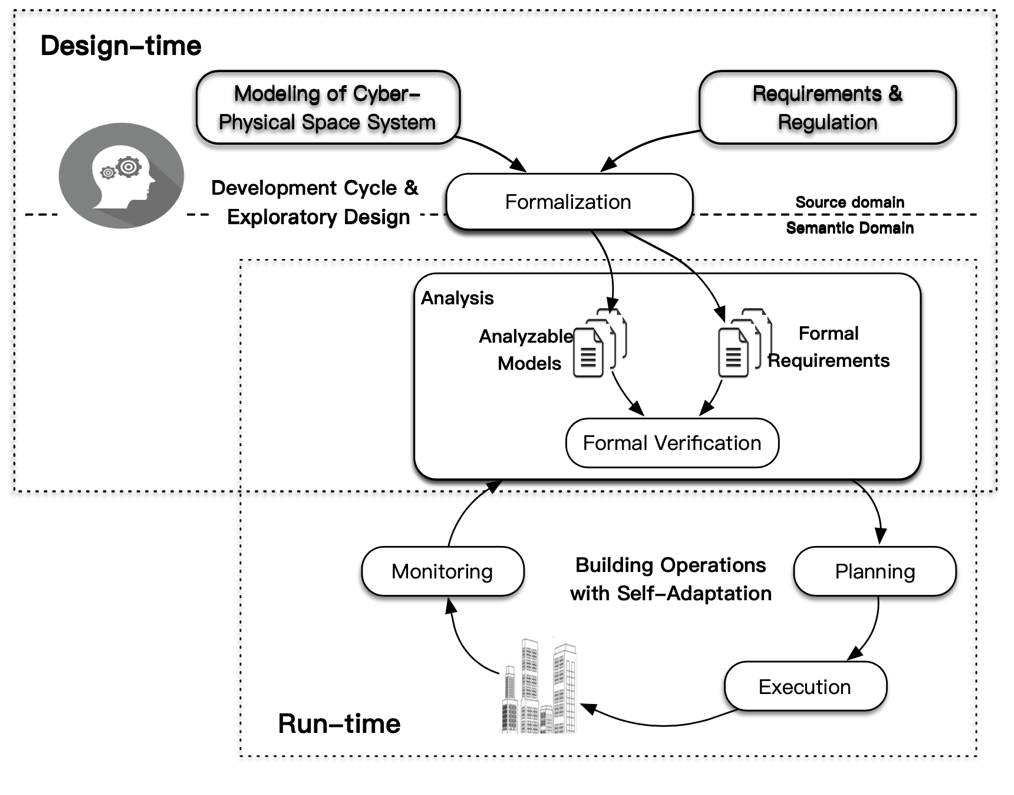
根据第五节中编写的系统性质形式化定义，在uppaal中实现了对应性质的检查语句，以下截图给出本项目所建立模型在uppaal中的验证结果。结果表明，实现的系统建模满足系统要求的所有性质定义，从而模型的正确性和有效性得以验证。



七、总结与感想

经过一学期以来对这门课程的学习，我们从一开始接触到的正则语言和有穷自动机，到之后的图灵机等等，非常详尽系统地学习了形式语言和自动机的理论知识。

在具体完成项目的实践过程中，让我对时间自动机建模验证工具uppaal有了深入的了解，通过阅读官方教学手册、学习学姐提供的PPT材料以及与同学进行讨论的过程，提高了我的自学能力以及快速定位问题、查找解决方案和解决问题的能力。尽管有关uppaal的相关资料比较有限，且难以查阅，但念语学姐提供的相关材料对我帮助很大。完成本课程项目则给了我一个很好的机会完整地体会一个真实系统，从最开始的设计，到具体细节实现，再到验证，以及发现问题后反复迭代完善调优模型的过程，让我很好地将理论与实践相结合起来，也对下面这张系统设计实现和验证的模式图加深了理解。



最后非常感谢金老师、杨老师以及整个课程团队一学期以来的辛勤劳动和努力，为我们呈现了精彩的课堂；精心设计的课程项目，也让我们能学有所得，学有所用。